Callable Objects Solutions

Copyright © James Raynard 2021

std::function

- What is the purpose of std::function?
  - std::function is a generic function object which can represent any type of callable object
  - std::function allows us to pass callable objects as arguments without having to write an overload for each kind of callable object
  - std::function also allows us to create containers of callable objects and populate them with assorted types of callable objects

Copyright © James Raynard 2021

Limitations of std::function

- Are there any limitations to std::function?
  - The type of the callable object is not stored
  - The parameter must exactly match the signature of the callable object
  - Invoking a callable object through std::function has runtime overhead, because the call is made indirectly
  - std::function may allocate memory on the heap, if the callable object is too large to be stored inside the std::function object
- What alternatives are there to using std::function?
  - For storing a callable object in a variable, use auto
  - This retains the type information
  - Invoking the callable object has no runtime overhead, because the call is made directly

Copyright © James Raynard 2021

count_strings

- Write a count_strings function similar to std::count_if which
  - Takes a vector of std::string and a function pointer
  - Calls the function on each element
  - If the function returns true, increments a counter
  - Returns the final value of the counter
- Write a program to test your code. It should work correctly with this function

```
bool match(const string& test) {
    return test == "two";
}
```

Copyright © James Raynard 2021

std::function

- Modify your code to use std::function instead of a function pointer. Check that it still works
- Modify the program to pass a lambda expression to your code instead of a function pointer. Check that it still works

Copyright © James Raynard 2021

std::bind

- Using this version of match and your count_strings function

```
bool match(const string& text, const string& value) {
   return text == value;
}
```

- Write a program to print out the number of occurrences of “two” in this vector

```
vector<string> texts = {"one", "two", "three", "two", "four", "two", "three"};
```

- Using a lambda expression with capture
- Using an object created with std::bind

Copyright © James Raynard 2021